Journal of KIIT. Vol. 17, No. 12, pp. 135-145, Dec. 31, 2019. pISSN 1598-8619, eISSN 2093-7571 135
http://dx.doi.org/10.14801/jkiit.2019.17.12.135

모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법

김하연*, 노혜민**, 유철중***

GUI Event Extraction Crawling Techniques for GUI Testing in Mobile Applications

Ha-Yeon Kim*, Hye-Min Noh**, and Cheol-Jung Yoo***

본 연구는 2019년 과학기술정보통신부 정보통신산업진흥원의 지역균형발전 SW ICT융합 기술개발(태양광 발전연계 지능형 축사 제어 ICT 디바이스 개발) 사업의 지원을 받아 수행된 연구임

요 약

최근 들어 스마트폰이 전 세계적으로 사용되면서 모바일 애플리케이션과 함께 새로운 기능들이 빠르게 생겨나고 있다. 이에 따라 애플리케이션의 기능 테스트가 중요하게 대두되고 있으며, 애플리케이션의 GUI를 테스트 하는 기법에 대한 연구가 많이 이루어지고 있다. 본 논문에서는 모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법을 제안한다. 이 기법은 먼저 애플리케이션을 실행하여 GUI 크롤링을 통해 이벤트를 추출하여 이벤트 리스트를 생성한 후, 입력 데이터를 설정하여 추출된 모든 액티비티의 이벤트 리스트를 실행하며 자동으로 테스트를 수행한다. 테스트 수행 결과 모든 액티비티가 자동으로 탐색되어 이벤트가 추출되었으며 경계 값과 동치 클래스 데이터를 입력하여 이벤트들이 실행됨을 확인하였다.

Abstract

Recently, smartphones are used worldwide. New features are rapidly emerging with mobile applications. As a result, functional testing of applications is becoming important, and a lot of research on application GUI testing techniques has been performed. This paper proposes a GUI event extraction crawling technique for mobile applications GUI testing. This technique first runs an application, extracts the events with a GUI crawling and generates an event list, set input data to run the event list of all extracted activities perform automatically tests. As a result of the test, all activities were automatically searched and the event was extracted and the event was executed by inputting the boundary value and equivalence class data.

Keywords

GUI testing, crawling, mobile application testing, GUI event extraction, input data setting

* 전자부품연구원 연구원
- ORCID: http://orcid.org/0000-0002-8427-6383
** 전북대학교 소프트웨어공학과 강의전담교수
- ORCID: http://orcid.org/0000-0003-1150-3570
*** 전북대학교 소프트웨어공학과 교수(교신저자), CAIIT
- ORCID: http://orcid.org/0000-0002-7903-2309

· Received: Nov. 05, 2019, Revised: Dec. 18, 2019, Accepted: Dec. 21, 2019
· Corresponding Author: Cheol-Jung Yoo
Dept. of Software Engineering, CAIIT, Jeonbuk National University,
567, Baekje-daero, Deokjin-gu, Jeonju-si, Jeollabuk-do, Korea,
Tel.: +82-63-270-3383, Email: cjyoo@jbnu.ac.kr

Ⅰ. 서 론

최근 몇 년 동안 스마트폰 안드로이드 애플리케이션의 수가 증가되면서 공공기관을 시작으로 다양한 분야의 모바일 서비스가 출시되고 있다[1]. 애플리케이션이 우리 삶의 모든 측면에서 사용되기 시작하면서 새로운 애플리케이션이 생겨나고 기능이 발전되고 있다. 기능이 발전함에 따라 애플리케이션 테스트가 개발 프로세스의 큰 부분을 차지하게 되었다. 애플리케이션이 설계 목적에 맞게 동작하고 있는지, 시스템 중단으로 인해 사용성에 불편함을 주지 않는지 테스트하는 것이 중요하다[2][3].

수동 테스트는 테스터가 직접 테스트 케이스를 작성하여 테스트를 수행한다. 직접 테스트 케이스를 작성하기 때문에 오류가 발생하기 쉽고 복잡한 애플리케이션의 모든 동작을 테스트하는 것은 기하급수적인 시간과 비용이 들어간다[4][5]. 안드로이드 애플리케이션의 복잡성은 그래픽 사용자 인터페이스(GUI)에 있다. GUI는 이벤트 기반으로 사용자가 애플리케이션 액티비티의 임의 지점을 다양하게 클릭하여 애플리케이션을 실행한다. 이러한 GUI 복잡성으로 인해 GUI 테스트는 큰 비용과 많은 시간이 요구된다[6][7]. 따라서 테스트를 실행할 때마다 조건을 동일하게하기 어렵기 때문에 GUI를 자동으로 테스트할 필요가 있다.

GUI 테스트는 화면을 구성하는 위젯을 클릭, 롱-클릭, 스크롤과 같은 사용자 이벤트를 통해 이벤트를 실행하여 프로그램이 올바르게 동작하는지 검증하는 기법이다[8]. GUI 테스트를 위해 많은 연구가 수행되고 있다. GUI 테스트를 자동화하기 위한 테스트 기법[9]-[11], GUI 위젯 기반 테스트, 오픈 소스를 사용한 테스트 기법 등 많은 테스트 방법이 제안되었다[12][13]. 주로 사용되는 GUI 테스트 방법은 Record Play-back 테스트, 랜덤 테스트, 모델 기반 테스트가 있다[14][15]. GUI 테스트에서 중요한 측면은 GUI 이벤트가 발견되고 실행되는 방법과 순서, 이벤트가 실행될 때 애플리케이션 및 실행 환경의 사전 조건, GUI의 탐색을 중지하는 데 사용되는 기준, 애플리케이션의 초기 상태가 있다[16]. 기존 테스트 기법들은 위에 설명된 GUI 특성을 고려하지 않아 GUI를 탐색하는 동안 애플리케이션의 런타임 충돌을 감지하는 데 어려움이 발생한다.

본 논문에서는 이러한 문제를 해결하기 위해 모바일 애플리케이션 GUI를 테스트하기 위한 자동으로 애플리케이션의 모든 액티비티를 방문하여 이벤트를 추출한 뒤 입력 데이터를 설정하여 이벤트를 실행시키는 GUI 크롤링 기법을 제안한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서 관련 연구로서 모바일 애플리케이션 GUI 테스팅 기법과 GUI 크롤링 기법에 대하여 기술한다. 3장에서 GUI 이벤트 추출 크롤링을 활용한 GUI 테스팅 기법에 대하여 기술한다. 4장에서 제안한 기법을 실제 애플리케이션에 적용한 사례를 제시한다. 마지막으로 5장에서 제안한 연구의 결론을 제시한다.

Ⅱ. 관련 연구

2.1 모바일 애플리케이션 GUI 테스팅

Woramet Muangsiri는 사용 로그를 기반으로 Behavioral Model을 생성하여 Behavioral Model과 수동 테스트를 결합한 행동 기반 랜덤 GUI 테스트를 제안하였다[17]. 사용 로그를 미리 수집하여 애플리케이션의 GUI 트리를 추출하고 통계 모델을 적용하여 초기 Behavioral Model을 만들어 GUI 모델에 매핑한 다음 Behavioral Model에서 임의의 이벤트를 선택하여 이벤트를 실행한다. 이벤트가 발생 하는 빈도별로 확률을 조정하여 필요한 경우 모델을 업데이트하고 제한한 시간에 도달할 때까지 과정을 반복하여 테스트가 진행될 때마다 Behavioral Model을 지속해서 업데이트가 가능하다. 실제 사용 로그를 기반으로 하여 초기 모델을 생성하기 위해 수동 테스트를 진행해야만 한다.

Cuixiong Hu는 JUnit을 사용하여 테스트 케이스를 생성한 뒤 Mokey 도구를 사용하여 이벤트를 생성하고 실행하여 버그를 찾는 방법을 제안하였다[18]. 애플리케이션의 소스 코드에서 시작하여 Java 테스트 케이스 생성도구인 JUnit을 사용하여 테스트 케이스를 생성하였다. 자동 이벤트 생성 도구인 Monkey를 사용하여 이벤트를 생성한 뒤, 랜덤으로

테스트 수행하였다. 테스트 케이스가 수행된 결과는 시스템 로그 파일에 기록되어 테스트 케이스가 실행된 후 잠재적인 버그를 탐지하기 위해 로그 파일 분석을 수행하였다. 생성된 테스트 케이스를 통해 랜덤으로 이벤트를 생성하기 때문에 API 오류나 I/O 오류를 발견하지 못한다.

Imparato는 GUI Ripping과 입력 데이터 교란 테스트 방법을 결합한 방법을 제안하였다[7]. GUI Ripping은 사용자의 별도 지시 없이 텍스트 필드에 일반적으로 로그인 화면에 유효한 사용자 아이디/암호와 같은 특정 입력 데이터를 제공하였다. Slum Droid를 사용하여 애플리케이션에서 실행 가능한 이벤트를 추출하고 입력 데이터 교환 방법을 수행하기 위해 개발된 GUIAnalyzer를 사용하여 테스트 케이스를 생성하였다. GUIAnalyzer은 텍스트 필드의 속성이나 이를 포함하는 GUI 이미지를 보여주는 도구로 입력 텍스트 유형과 관련된 정규식을 교란하여 텍스트 입력을 생성시켜 입력 목록을 XML 형식으로 저장해준다.

2.2 크롤링을 활용한 GUI 테스팅

Salihu는 모바일 애플리케이션에서 역공학 모델에 대한 하이브리드 정적-동적 접근 방식으로 UI 모델을 자동으로 생성하기 위한 도구 AMOGA를 제안하였다[19]. AMOGA는 모바일 애플리케이션의 바이트 코드에 대한 정적 분석을 수행하여 이벤트 추적 알고리즘을 통해 실행 순서에 따라 이벤트 집합을 생성하였다. 동적 분석을 위한 입력으로 사용될 수 있는 애플리케이션의 이벤트는 동적 크롤러를 사용하여 추출한다. 추출된 이벤트들은 체계적으로 실행되어 State 모델을 탐색하고 AMOGA는 가장 짧은 UI 경로를 탐색하였다. 제안된 방법은 가장 짧은 UI 경로를 탐색하기 때문에 모든 경로를 탐색할 수 없는 위험이 존재한다.

Domenico가 제안한 방법으로 애플리케이션의 UI를 체계적으로 리핑하는 Ripper를 사용하여 애플리케이션을 동적 분석하여 코드에서 결함을 발생시킬 수 있는 테스트 케이스를 생성하는 Android Ripper가 있다[16]. AndroidRipper는 GUI 위젯에서 실행 가능한 이벤트 시퀀스를 얻는 것을 목표로 하여 애플리케이션을 동적으로 분석하였고, 발생 가능한 GUI 상태와 상태 전이를 포함한 GUI 모델을 생성하였다. AndroidRipper는 테스트 케이스를 자동으로 생성하고 실행하지만 재사용 가능한 모델을 개발하지 않다는 것과 임의의 데이터를 사용하기 때문에 특정 경우의 테스트 케이스만 고려되는 문제가 있다. Domenico가 제안한 다른 방법으로 GUI에서 발생할 수 있는 실제 사용자 이벤트를 실행하여 크롤러를 기반으로 GUI 모델을 자동으로 생성하는 방법이 있다[20]. 크롤러는 실행 중인 액티비티, 액티비티가 구현하는 이벤트 핸들러, 액티비티에 포함된 위젯 정보를 추출하여 이벤트를 실행하였다. 실행 중 런타임 충돌 오류와 같은 결함을 찾아내거나 회귀 테스트에 사용될 수 있는 테스트 케이스를 생성하였다. 입력 데이터는 임의의 데이터 값만으로 테스트를 진행하기 때문에 생성된 테스트 케이스는 사전 조건에 대한 정보가 반영되지 않는다.

Memon et al은 실행 중인 GUI에서 직접 GUI 애플리케이션의 역공학 모델을 제공하는 GUITAR를 제안하였다[21]. GUITAR은 포리스트 형식과 이벤트 흐름 그래프(EFG)를 사용하여 GUI의 구조와 실행 가능한 이벤트를 표현한다. 생성된 EFG는 일반적으로 잘못된 이벤트 시퀀스가 포함되어 있어 나중에 제거해야 할 수도 있다.

Wei Yang은 모바일 애플리케이션의 모델을 자동으로 추출하기 위한 Gray-Box 접근 방식을 제안하였다[22]. 애플리케이션의 소스 코드를 정적 분석을 사용하여 GUI의 위젯이 지원하는 사용자 이벤트 집합을 추출하고 동적 크롤러를 사용하여 추출된 이벤트를 체계적으로 실행하여 모델을 생성하였다. 제안된 방법은 GUI를 시각적으로 관찰 가능한 상태로 구성하기 위해 실행 가능한 구성 요소의 속성을 수동으로 선택해야 한다.

2.3 제안한 GUI 이벤트 추출 크롤링 기법

본 논문에서는 위에서 언급한 기존 연구들의 문제를 해결하기 위해 모바일 애플리케이션 GUI를 테스트하기 위한 GUI 이벤트 추출 크롤링 기법을

제안한다. 이 기법은 APK 분석을 통해 입력 데이터로 사용할 수 있는 경계 값과 동치 클래스 데이터를 Config 파일에 저장한 뒤 애플리케이션을 실행한다. 자동으로 실행한 애플리케이션의 모든 액티비티를 방문하여 이벤트를 추출하고 Config 파일에 설정한 데이터를 입력 데이터로 설정하여 이벤트를 실행하는 GUI 크롤링 기법을 제안한다.

III. 모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법

본 장에서는 제안한 모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법을 제안한다.

3.1 제안한 기법의 개요

제안한 기법은 그림 1의 개요도를 보면 크게 APK 분석, 애플리케이션 GUI 테스팅과 같이 2가지 단계로 구성된다.APK 분석 단계는 안드로이드 APK 파일을 디컴파일하여 바이트 코드와 레이아웃 XML 파일을 얻어낸 후 생성된 바이트 코드와 레이아웃 XML 파일은 결합하여 안드로이드 프로젝트를 생성한다. 생성된 안드로이드 프로젝트에서 이벤트의 입력 데이터로 사용할 수 있는 경계 값과 동치 클래스 데이터를 확인하여 Config 파일에 저장한다. GUI 크롤링 단계는 애플리케이션을 실행하여 크롤링 알고리즘을 수행한다. UI Automator API를 사용하여 애플리케이션의 현재 액티비티에서 이벤트를 추출하여 이벤트 리스트를 생성한다. 입력 데이터가 필요한 이벤트의 경우 Config에 저장한 데이터를 입력 데이터로 설정하여 이벤트를 실행한다. 모든 액티비티를 방문하고 이벤트 리스트에 저장된 이벤트가 모두 수행될 때까지 크롤링 알고리즘을 반복한다.

3.2 APK 분석 단계

APK(Android Application Package)란 안드로이드 소프트웨어와 미들웨어 배포에 사용되는 패키지 파일로, 안드로이드용 프로그램을 컴파일하여 생성한다. 프로그램의 소스 코드와 Resource, information, 매니페스트 파일 등을 포함하고 있기 때문에 APK 파일을 디컴파일하면 애플리케이션의 소스 코드 정보를 볼 수 있다.

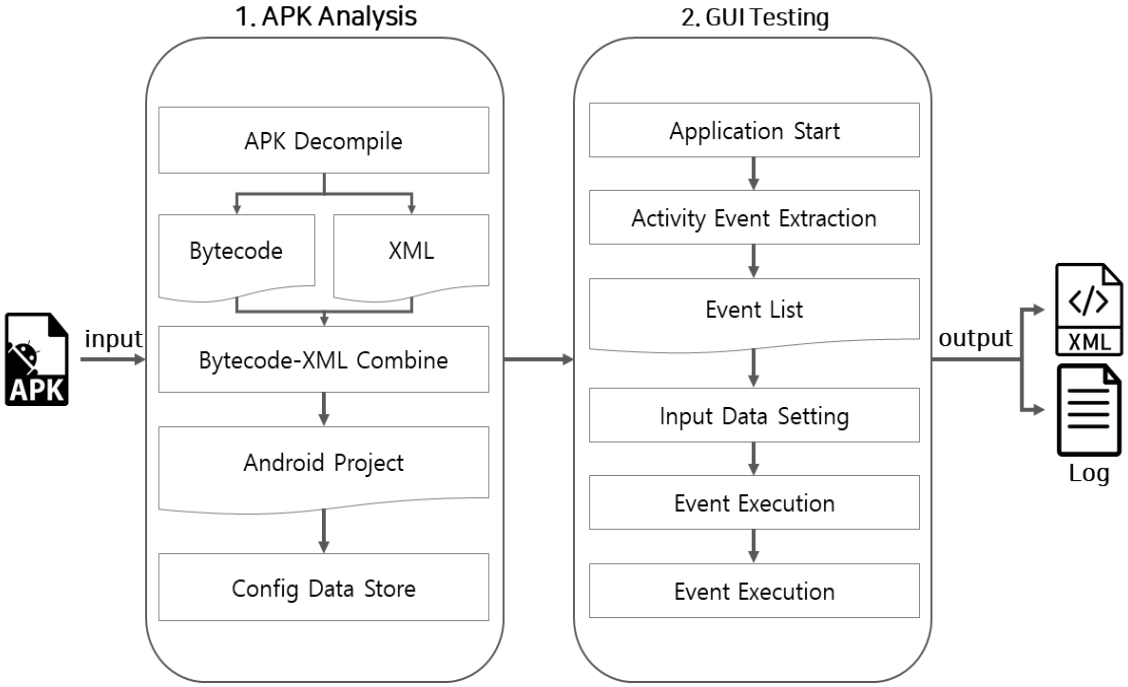

그림 1 . 제안한 모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법
Fig. 1. A GUI event extraction crawling technique for mobile application GUI testing

APK 파일은 Bytecode Viewer를 사용하여 디컴파일을 수행할 수 있다. Bytecode Viewer로 APK 파일을 바이트 코드와 클래스 파일로 변환해서 소스 파일로 복원된 결과를 볼 수 있다. 하지만 Bytecode Viewer는 자바 파일이나 class 파일은 변환할 수 있으나 레이아웃이나 매니페스트에 사용되는 XML 파일은 변환이 불가능하다는 단점이 있다. 이러한 단점은 APKtool을 사용하여 보완할 수 있다. APKtool의 디컴파일 결과는 파일로 생성되며, 변환된 XML 파일을 얻을 수 있다. Bytecode Viewer를 통해 얻은 소스 파일과 APKtool을 통해 얻은 XML 파일을 결합하여 안드로이드 프로젝트 파일을 생성한다. 소스 코드에서 입력 데이터로 사용할 수 있는 경계 값과 동치 클래스 데이터를 확인하여 Config 파일변수에 저장한다.

동치 클래스는 입력 값을 동일한 결과가 예상되는 영역으로 나누어 각 영역에 해당하는 대표 값을 말하며, 경계 값은 입력 데이터가 유효에서 무효로 바뀌는 경계에 있는 값을 말한다. 이벤트의 입력 데이터는 방대하기 때문에 모든 입력 데이터를 조합하면 테스트 케이스 수가 기하급수적으로 늘어난다. 경계 값과 동치 클래스를 입력 데이터로 사용함으로써 데이터 조합 수를 작게 유지하면서 입력 이벤트의 결함 발견율을 높일 수 있다.

Config 파일은 그림 2와 같이 크롤링을 실행하기 위해 필요한 환경을 설정하는 데이터를 저장한 클래스 파일로, 경계 값과 동치 클래스 데이터를 확인하여 TEXT 변수에 데이터를 입력한다.

```
public class Config {
    public static String targetPackage =  "";
    public static String basicPackage = "com.crawler";

    public static int maxDepth = 25;
    public static int maxSteps = 999;

    public static File outputDir;
    public static String fileLog;

    public static final String[] TEXT = {

    };
```

그림 2. Config 파일 데이터
Fig. 2. Config file data

3.3 GUI 크롤링 단계

이 단계에서는 애플리케이션을 실행하여 자동으로 액티비티를 탐색하여 GUI 이벤트를 추출한 뒤 입력 데이터를 설정하여 이벤트를 실행하는 크롤링 알고리즘 수행한다.

테스트 중인 애플리케이션(AUT)에서 GUI 테스트를 수행하기 위해서는 우선 AUT의 액티비티에서 실행 가능한 이벤트를 추출해야 한다. 이벤트는 클릭, 롱-클릭, 터치 등과 같은 이벤트를 발생시키는 액티비티의 위젯에서 정보를 얻을 수 있다. UI Automator API를 사용하여 AUT의 액티비티 정보와 위젯의 이벤트 정보를 추출할 수 있다. UI Automator는 시스템 및 설치된 애플리케이션의 UI 테스트에 적합한 UI 테스트 프레임워크로써 UI 테스트를 할 수 있도록 API를 제공한다. 표 1은 UI Automator API를 나타낸다.

UiDevice를 사용하여 애플리케이션이 실행되는 장치 정보와 실행 중인 액티비티에 대한 정보를 가져온다. 실행 중인 액티비티는 activityList에 정보를 저장하고, UiObject와 UiSelector를 사용하여 액티비티에서 발생 가능한 위젯 이벤트를 얻을 수 있다.

표 1. UI Automator API 종류
Table. UI Automator API types

API	Function
UiCollection	Used to enumerate a container's UI elements for the purpose of counting, or targeting a sub elements by a child's text or description.
UiObject	A representation of a view. It is not in any way directly bound to a view as an object reference
UiScrollable	A UiCollection and provides support for searching for items in scrollable layout elements
UiSelector	Specifies the elements in the layout hierarchy for tests to target, filtered by properties such as text value, content-description, class name, and state information
UiDevice	provides access to state information about the device

본 논문에서는 표 2와 같은 위젯을 추출한다.

표 2. 추출되는 위젯 종류
Table 2. Extracted widget type

Widget	Function
TextView	A user interface element that displays text to the user.
EditText	A user interface element for entering and modifying text.
ListView	Displays a vertically-scrollable collection of views
Spinner	A view that displays one child at a time and lets the user pick among them
Button	A user interface element the user can tap or click to perform an action.
CheckBox	A specific type of two-states button that can be either checked or unchecked
RadioButton	A two-states button that can be either checked or unchecked
ToggleButton	Displays checked/unchecked states as a button with a "light" indicator and by default accompanied with the text "ON" or "OFF".
ImageButton	Displays a button with an image (instead of text)

UiSelector를 사용하여 clickable, longclick, scrollable 등과 같은 이벤트를 지니는 위젯들을 추출한다. activityList와 widgetList는 실행 여부를 확인하기 위해 boolean 타입의 'finished' 변수를 함께 저장한다. 기본은 false 상태이며, 위젯이 수행되면 true 데이터를 저장하며, 추출된 위젯들은 UiObjcet 형태의 widgetList에 저장한다.

생성된 widgetList의 이벤트를 실행하기 위해서는 UI Automator API 중에서 UiObject에서 제공하는 기능으로 위젯을 실행할 수 있다. UiSelector은 className, text, resourceId 정보를 검색 조건으로 사용하여 위젯을 실행한다. className, text, resourceId 정보는 앞에서 추출한 이벤트에서 정보를 얻어올 수 있다. Button은 click 메소드를 사용하여 버튼 클릭을 실행하며, Textview나 EditText는 setText 메소드를 사용하여 텍스트 입력을 실행한다. Checkbox는 isCheckable 메소드를 사용하여 체크 여부를 판단하여 이벤트를 실행한다. 이 외에도 UiObject에서 제공하는 메소드를 사용하여 롱-클릭, 드래그, 스와이프, 스크롤 등의 이벤트를 실행할 수 있다.

이벤트는 입력 데이터가 필요한 이벤트와 데이터 입력 없이 실행 가능한 이벤트가 있다. 입력 데이터가 필요한 이벤트의 경우 이벤트를 실행하기 전에 Config에 저장된 경계 값과 동치 클래스 데이터를 입력하여 데이터를 설정하여 이벤트를 실행한다.

이벤트를 실행하면 실행된 위젯의 finished 상태를 true로 변경한다. 이벤트 실행 결과가 충돌 없이 실행되면 유효한 데이터로 식별하여 데이터 유효성에 valid를 저장하고 입력된 데이터를 함께 저장한다.

Algorithm

```
Input : App
Output : XML file, Logfile

start(App)
a ← App.activity
finished ← false
activityList : activityList
widgetList : widgetList
w : widget

while finished ≠ true
    if a = newActivity
        activityList ← a
        widgetList ← getWidgetList
    end if
    input data setting
    if widgetList.isFinished ≠ finish
        if widget.activity ≠ currentActivity
            backtrackToPrevious
        endif
    endif
    elseif widgetList.isFinished = finish
        remove eventList
        nextActivity
    endif
    if activityList.isFinished = finish
        remove eventList
        finished ← true
    endif
    elseif activityList.isFinished ≠ finish
        continue
    endif
end
```

그림 3. GUI 크롤링 알고리즘
Fig. 3. GUI crawling algorithm

충돌이 발생하면 데이터 유효성에 invalid를 저장하고 입력된 데이터를 함께 저장한다. 입력 이벤트가 필요하지 않은 이벤트는 유효성에 invalid와 입력 데이터는 null을 입력하여 데이터를 저장한다. 로그 파일에 기록되는 Exception으로 데이터 유효성과 오류를 판단한다.

widgetList에 저장된 모든 위젯의 finished 상태가 true가 될 때까지 알고리즘을 수행한다. widgetList의 위젯 상태가 모두 true가 되었을 경우 액티비티의 finished 상태를 true로 변경한다. false 상태인 액티비티가 존재하면 해당하는 액티비티로 돌아가서 실행되지 않은 위젯의 이벤트를 수행한다. 이러한 과정을 반복하여 애플리케이션의 activityList와 widgetList의 finished 상태가 모두 true가 될 때까지 크롤링 알고리즘을 반복하면서 이벤트를 수행한다. 그림 3은 GUI 크롤링 단계의 알고리즘을 설명한다.

Ⅳ. 적용사례

본 장에서는 제안된 GUI 크롤링 기법을 실제 애플리케이션에 적용한 사례를 기술한다. 실험에는 'Tomdroid'애플리케이션을 사용하였다. Tomdroid는 데이터를 저장하고 읽을 수 있는 메모 기능을 제공하는 애플리케이션이다.

첫 번째 APK 분석 단계에서 APK 파일을 Bytecode Viewer를 사용하여 디컴파일을 수행한다. 디컴파일 결과는 그림 4와 같이 볼 수 있다. XML 파일을 변환하기 위해 APKtool을 사용하여 변환한 결과는 그림 5와 같이 수행되어 XML 파일을 반환한다.

두 과정의 결과를 결합하여 안드로이드 프로젝트를 생성한다. 생성된 소스 코드에서 경계 값과 동치 클래스 데이터로 사용할 수 있는 입력 값이 있는지 확인하여 Config 파일의 TEXT 변수에 데이터를 입력한다. 그림 6은 Config 파일에 저장된 TEXT 변수를 나타낸다.

애플리케이션을 실행하여 GUI 크롤링 기법을 적용하여 이벤트를 추출한다. 입력 데이터가 필요한 이벤트는 Config 파일의 TEXT 데이터를 사용하여 입력 데이터를 설정하여 이벤트를 실행한다. 모든 액티비티를 탐색하고 추출한 이벤트 리스트의 이벤트가 모두 실행될 때까지 Tomdroid 애플리케이션의 이벤트를 반복적으로 실행한다. 테스트를 수행한 결과 애플리케이션이 자동으로 액티비티를 방문하여 이벤트를 실행하였고 수행 중 그림 7과 같은 Crash 에러가 발생함을 확인할 수 있었다.

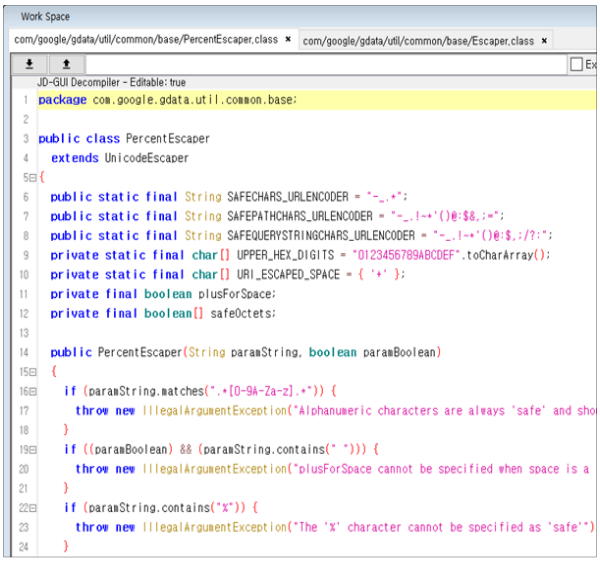

그림 4. BytecodeViewer 실행 결과
Fig. 4. BytecodeViewer execution result

그림 5. APKtool 실행 결과
Fig. 5. APKtool execution result

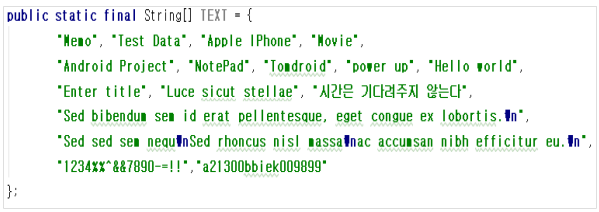

그림 6. Config 파일 - TEXT 데이터
Fig. 6. Config file - TEXT data

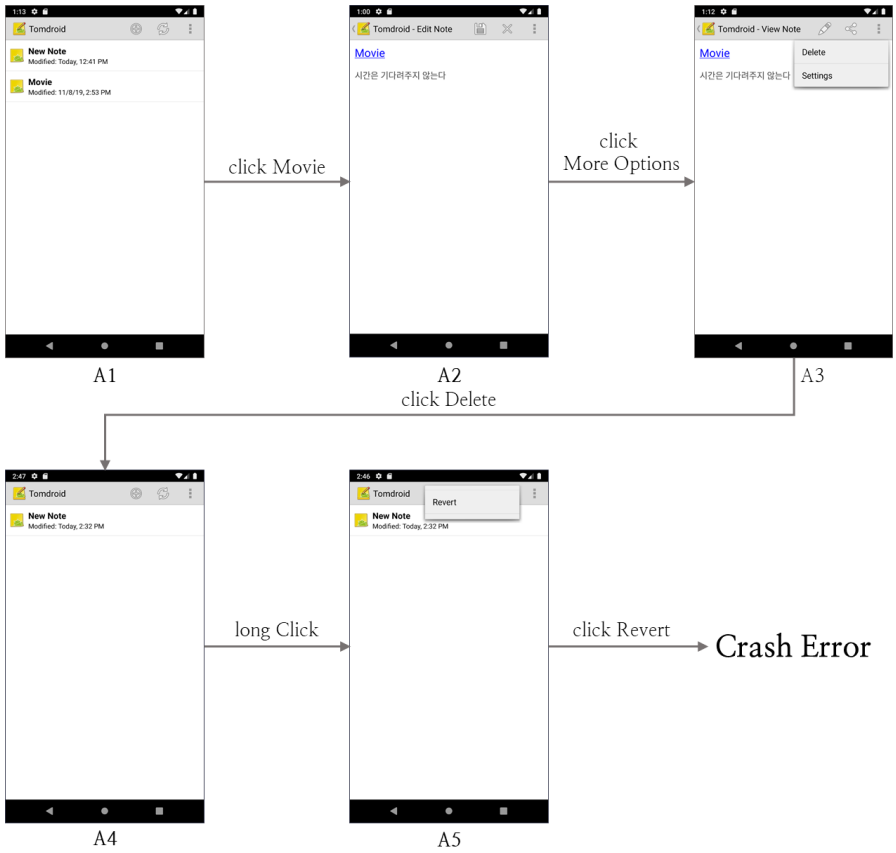

그림 7. Tomdroid 에러 흐름
Fig. 7. Tomdroid error flow

```
<?xml version="1.0" encoding="UTF-8"?>
<Graph>
    <screenNode id="1" depth="0" activityName="Tomdroid"
    pakageName="org.tomdroid"
    signature="FrameLayout;ViewGroup;FrameLayout;ViewGroup;
    LinearLayout;FrameLayout;ImageView;LinearLayout;TextView;
    LinearLayout;TextView;LinearLayout;TextView;">
        <screenEdge message="{Click} LinearLayout" bounds="[21,63][349,189]"
        dataType="invalid" inputData="null" nextActivity="1"/>
        <screenEdge message="{Click} More options ImageButton"
        bounds="[932,63][1080,189]" dataType="valid"
        inputData="null" nextActivity="1"/>
    </screenNode>
    <screenNode id="2" depth="1" activityName="Tomdroid"
    pakageName="org.tomdroid" signature="FrameLayout;FrameLayout;ListView;
    LinearLayout;LinearLayout;RelativeLayout;TextView;LinearLayout;
    LinearLayout;RelativeLayout;TextView;LinearLayout;LinearLayout;
    RelativeLayout;">
        <screenEdge message="{Click} LinearLayout" bounds="[544,189][1059,315]"
        dataType="valid" inputData="null" nextActivity="2"/>
    </screenNode>
```

그림 8. Tomdroid XML 결과
Fig. 8. Tomdroid XML result

그림 8은 GUI 크롤링 알고리즘의 결과를 담고 있는 XML 파일의 내용을 보여준다. 애플리케이션을 실행하면서 발생한 액티비티의 정보와 위젯의 이벤트 정보를 나타내고 있다. XML파일에서 screenNode 정보는 탐색한 액티비티의 id와 깊이, 패키지 명, signature 정보가 저장되었음을 확인할 수 있으며, 액티비티의 노드 안에는 이벤트 정보를 담고 있는 screenEdge가 있다. screenEdge는 액티비티에서 발생한 이벤트의 정보가 저장되며, 저장된 이벤트 정보는 이벤트를 발생시킨 위젯의 이름, 액티비티에서 위젯의 위치를 나타내는 bounds 정보, 입력 데이터, 데이터의 유효성, 이벤트 발생 후 디바이스에 보여지는 액티비티 id가 있다. 데이터의 유효성은 Crash 에러가 발생할 경우 invalid로 표기된다. 데이터의 유효성을 보고 에러가 발생한 액티비티와 발생한 이벤트인지 어떠한 데이터 형태가 에러를 발생시키는지 확인할 수 있었다. XML 파일은 액티비티와 이벤트의 계층 구조로 데이터를 저장하고 있기 때문에 FSM 모델과 GUI 트리 모델 등으로 응용하여 여러 모델을 생성할 수 있어 모델 기반 테스트에 적용할 수 있다.

V. 결 론

본 논문에서는 모바일 애플리케이션의 APK를 분석하여 경계 값과 동치 클래스 데이터를 변수로 저장하고, GUI 크롤링을 통해 자동으로 액티비티를 탐색하여 이벤트를 추출하고 입력 데이터를 설정하여 이벤트를 실행하는 GUI 크롤링 기법을 제안하였다. 제안한 방법을 실험하기 위해 실제 배포되고 있는 안드로이드 애플리케이션에 적용한 결과, 크롤링 기법을 사용하여 GUI 이벤트가 자동으로 실행되었고 Crash 에러를 발견함을 확인하였다. 또한 잘못된 데이터 입력으로 인해 예기치 못한 런타임 오류와 같은 애플리케이션의 동작에 영향을 주는 이벤트 버그를 발견하였다.

기존 연구들과 달리 임의의 값이 아닌 경계 값과 동치 클래스 데이터를 입력 데이터로 설정하여 입력 조합의 수를 감소시켰고 자동으로 애플리케이션의 모든 액티비티를 방문하여 발생 가능한 이벤트를 추출하면서 실행하기 때문에 GUI가 변경되더라도 알고리즘을 재사용 가능하여 테스트 비용과 시간을 절감시킬 수 있을 것으로 보인다.

References

[1] Joo-Hee Lee, "UI Design Survey of Mobile App for the Activation of Traditional Market", The Journal of Institute of Internet Broadcasting and Communication(IIBC), Vol, 15, No. 1, pp. 277-282, Feb. 2015.

[2] Linus Esbjörnsson, "ANDROID GUI TESTING : A Comparative Study of Open Source Android GUI Testing Frameworks", Bachelor Degree Project in Informatics, pp. 1-48, 2015.

[3] Chien-Hung Liu and Ping-Hung Chen, "A Crawling Approach of Hierarchical GUI Model Generation for Android Applications", Journal of Internet Technology, Vol. 19, No. 5, pp. 1613-1623, Jan. 2018.

[4] Ilkka Kortelainen, "Automated GUI Testing for Android Applications", Bachelor's Thesis in Information Technology, pp. 1-18, Dec. 2012

[5] Beomjin Jeong, Jungwoo Lee, Changwan Hong and Beongku An, "GUI-based Black Box Test Automation Program Tool in Windows Environment", The Journal of Institute of Internet Broadcasting and Communication, Vol, 18, No. 2, pp. 163-168, Apr. 2018.

[6] Wontae Choi, Koushik Sen, George Necula, and Wenyu Wang, "DetReduce: Minimizing Android GUI Test Suites for Regression Testing", ACM/IEEE 40th International Conference on Software Engineering, pp. 445-455, May 2018.

[7] Gennaro Imparato, "A Combined Technique of GUI Ripping and Input Perturbation Testing for Android Apps", IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 2, pp. 760-762, May 2015.

[8] Youngmin Baek, Gwangui Hong, Cheonghyun Lee, and Doo-Hwan Bae, "A Screen Comparison Technique for Effective Model-based Android GUI Testing", Journal of KIISE, Vol. 42, No. 11, pp. 1386-1396, Nov. 2015.

[9] Davi Bernardo Silva, Andre Takeshi Endo, Marcelo Medeiros Eler, and Vinicius H. S. Durelli, "An analysis of automated tests for mobile Android applications", XLII Latin American Computing Conference, Oct. 2016.

[10] Ana C. R. Paiva, Joao M. E. P. Gouveia, Jean-David Elizabeth, and Marcio E. Delamaro, "Testing When Mobile Apps Go to Background and Come Back to Foreground", IEEE International Conference on Software Testing, Verification and Validation Workshops, pp. 102-111, Apr. 2019.

[11] Ahmed Mateen and Khizar Abbas, "Optimization of Model based Functional Test Case Generation for Android Applications", IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, pp. 90-95, Sep. 2017.

[12] Riccardo Coppola, Maurizio Morisio, and Marco

Torchiano, "Maintenance of Android Widget-based GUI Testing:A Taxonomy of test case modification causes", IEEE International Conference on Software Testing Verification and Validation Workshops, pp. 151-158, Apr. 2018.

[13] Riccardo Coppola, Maurizio Morisio, and Marco Torchiano, "Mobile GUI Testing Fragility: A Study on Open-Source Android Applications", IEEE Transactions on Reliability, Vol. 68, pp. 67-90, Mar. 2019.

[14] Mohammed lafi, Mohamed S. Osman, and Hiba Ayyed Wasmi, "Improved Monkey Tool for Random Testing in Mobile Applications", IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, pp.6 58-662, Apr. 2019.

[15] Amit Seal Ami, Rayhanur Rahman, Kazi Skib, and Md. Mehedi Hasan, "MobiCoMonkey-Context Testing of Android Apps", ACM/IEEE 5th International Conference on Mobile Software Engineering and Systems, pp. 76-79, May 2018.

[16] Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana, Salvatore De Carmine, and Atif M. Memon, "Using GUI Ripping for Automated Testing of Android Applications", IEEE/ACM International Conference on Automated Software Engineering, pp. 258-261, Sep. 2012.

[17] Woramet Muangsiri and Shimgo Takada, "Random GUI Testing of Android Application Using Behavioral Model", International Conference on Software Engineering and Knowledge Engineering, Vol. 27, No. 09n10, pp. 1603-1612, Dec. 2017.

[18] Cuixiong Hu and Iulian Neamtiu "Automated GUI Testing for Android Applications", AST '11 Proceedings of the 6th International Workshop on Automation of Software Test pp. 77-83, May. 2011.

[19] Ibrahim-Anka Salihu, Rosziati Ibrahim, Bestoun S. Ahmed, Kamal Z. Zamli, and Asmau Usman, "AMOGA: A Static-Dynamic Model Generation Strategy for Mobile Apps Testing", IEEE Access, Vol. 7, pp. 17158-17173, Jan. 2019.

[20] Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana, "A GUI Crawling-based technique for Android Mobile Application Testing", IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp. 252-261, Mar. 2011.

[21] Atif Memon, I. Banerjee, and A. Nagarajan, "GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing", Proceedings of the 10th Working Conference on Reverse Engineering, p. 260, Nov. 2003.

[22] Wei Yang, Mukul R. Prasad, and Tao Xie, "A Grey-Box Approach for Automated GUI-Model Generation of Mobile Applications", Proceedings of the 16th international conference on Fundamental approaches to software engineering, pp. 250-265, Mar. 2013.

저자소개

김 하 연 (Ha-Yeon Kim)

2016년 2월 : 전북대학교 소프트웨어공학과(공학사)
2016년 1월 ~ 현재 : 전자부품연구원 연구원
2017년 3월 ~ 현재 : 전북대학교 소프트웨어공학과(공학석사과정),
관심분야 : 소프트웨어 공학, 소프트웨어 품질 및 테스팅

노 혜 민 (Hye-Min Noh)

2000년 : 전북대학교
컴퓨터과학과(이학사)
2002년 : 전북대학교 대학원
전산통계학과(이학석사)
2006년 : 전북대학교 대학원
컴퓨터통계정보학과(이학박사)
2011년 3월 ~ 현재 : 전북대학교
소프트웨어공학과 강의전담교수
관심분야 : 소프트웨어공학, 소프트웨어 아키텍처,
소프트웨어 테스팅, HCI, 정형 명세 기법, 사물인터넷,
빅데이터, 기계학습, 정밀농업 등

유 철 중 (Cheol-Jung Yoo)

1982년 2월 : 전북대학교
전산통계학과(이학사)
1985년 8월 : 전남대학교 대학원
계산통계학과(이학석사)
1994년 8월 : 전북대학교 대학원
전산통계학과(이학박사)
2012년 1월 ~ 2013년 7월 :
University of California, Irvine(UCI) 국외연구교수
1997년 ~ 현재 : 전북대학교 소프트웨어공학과 교수
관심분야 : 소프트웨어 품질/메트릭스/테스팅, 임베디드
소프트웨어/테스팅, 빅데이터 분석, 머신러닝,
정밀농업 등